2023 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)
June 28-30, 2023. Seattle, Washington, USA

A Parameterized Cubic Bézier Spline-based Informed RRT* for Non-holonomic Path Planning

Zifan Fei, *Student Member, IEEE* and Ya-Jun Pan, *Senior Member, IEEE*

***Abstract*— This paper proposes a new path planning algorithm for robotics called Informed SRRT*. Compared to conventional RRT* algorithms with Euclidean metrics, our algorithm extends the approach by incorporating a local planner from SRRT to satisfy both external and internal constraints. To compute the path to the goal region, we use parameterized cubic curves instead of computationally expensive numerical methods. We add two extra lines at the endpoints of the Bézier spline to leave rooms for the rewiring process. Kinematic constraints require at least three state connections to be tweaked during rewiring. The algorithm always ensures that the path has G^2 continuity of curvature within upper-bound constraints. Simulation results demonstrate that the proposed method finds shorter paths than SRRT while maintaining the same iteration of node sampling.**

I. INTRODUCTION

Motion planning is a critical task in robotics with applications in self-driving cars [1], autonomous surface vehicles (ASVs) [2], and unmanned aerial vehicles (UAVs) [3]. It is also widely used in other areas, such as manufacturing, biology, and computing industry [4]. Recent research has focused on improving the feasibility [5, 6] and optimality [7–9] of paths, as well as achieving real-time applications with replanning processes [10, 11]. Motion planning is typically divided into two parts: global planning, which plans ahead to avoid static obstacles, and local planning, which handles the robot's nonholonomic constraints in real-time.

Global planners use different methods to discretize the continuous state space. Graph based searches such as D* lite [12] build a graph representation of the environment and search for a path. However, graph-based planners can suffer from the curse of dimensionality, which occurs when the number of states increases exponentially with the number of dimensions, making it difficult to build a graph representation in high-dimensional spaces. Sampling-based planners such as RRT [13] randomly sample the state space and attempt to connect nearby samples. RRT can find a path quickly in a large-size map due to Voronoi bias property and has the ability to consider the nonholonomic constraints at the same time, although it may not guarantee to find a strictly complete solution. In 2010, it is proved that RRT almost surely produces a suboptimal path [14]. To address this limitation, the authors proposed RRT*, which incorporate optimal strategies and gain huge popularity in the motion planning community in recent years. It guarantees asymptotic optimality, meaning that as the number of samples approaches infinity, the solution will converge to the optimal path. There are also several improved versions of RRT* such as RRT*-Smart [15], RRT$^{\#}$ [7] and informed RRT* [8].

Nonholonomic motion planning is a type of motion planning that takes into account kinematic and dynamic constraints such as the robot's curvature and maximum turning angles, which cannot be modeled with Euclidean distance alone. Therefore, using Euclidean distance as a metric to estimate effective distance for nonholonomic motion planning may not be precise. More detailed explanation can be seen in [16, 17]. In response to these challenges, a number of algorithms have been developed that can address the issue of nonholonomic motion planning. One such algorithm is CBB-RRT* [16]. CBB-RRT* is a nice algorithm that has been designed to rearrange the four control points and the flexible link point between two control states.

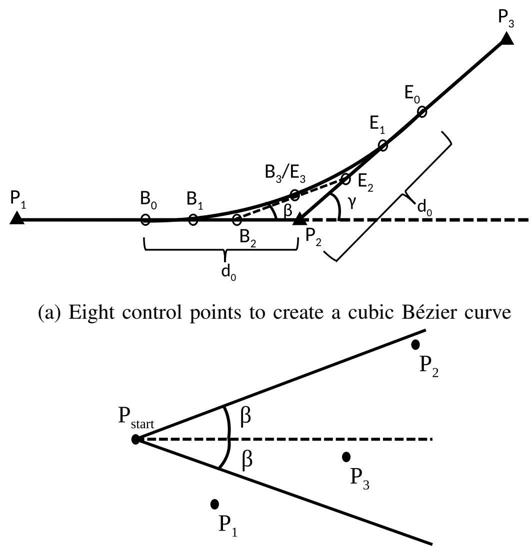
Inspired by [7, 8, 17], our method ensures that the node expansion maintains the feasibility check at all times. In 2D scenarios, it uses the direct sampling method from an admissible ellipse and modifies the rewire processes from RRT* with additional kinematic constraints. Compared to the straight line connection, our method needs to re-examine the feasibility among three states. If the rewired nodes has its own child nodes, then at least four consecutive states needs to be reconnected or deleted.

II. PROBLEM FORMULATION

The motion planning problem is described similar as [7, 8] did. Let $\mathcal{X} \subseteq \mathbb{R}^n$ denote the configuration state space, where $n \subseteq \mathbb{N}$ with $n \geq 2$. $\mathcal{X}_{obs} \subsetneq \mathcal{X}$ be the states in collision with obstacles. $\mathcal{X}_{free} = cl(\mathcal{X} \backslash \mathcal{X}_{obs})$ is the non-collision permissible state space that could be sampled and expanded for the spanning graph. $cl(\cdot)$ represents the closure of a set. $\mathbf{x}_{start}$ and $\mathbf{X}_{goal} \subset \mathcal{X}_{free}$ are the known initial point and goal region. Let $\mathcal{T} = (V, E)$ where V and E are the finite vertices and edges. Let $\sigma : [0, 1] \rightarrow \mathcal{X}_{free}$ becomes all the sequence of states that produce a continuous path for the nonholonomic robot, where $\sigma(0) = \mathbf{x}_{start}$ and $\sigma(1) \in \mathbf{X}_{goal}$. In [17], the new extended segment towards the feasible point will be generated by the analytical algorithm that always below the upper bound curvature constraints. There are eight control points (B_{0-3}, E_{0-3}) to generate a continuous curvature among three points (W_{1-3}), as shown in Fig 1(a).

Zifan Fei and Ya-Jun Pan are with the Department of Mechanical Engineering, Dalhousie University, Halifax, Canada, B3H 4R2. (e-mail: felix.fei@dal.ca ; yajun.pan@dal.ca).

The second constraint is a region bounded by the robot's maximum turning angles ranged from β_{max} and β_{min}. In Fig 1(b), P_{start} is the initial position of the robot and the dash line represents its direction. P_1 is not a feasible point even though it has the shortest distance compared to P_2 and P_3. Both P_2 and P_3 are feasible points for potential extension since they are within the ranges bounded by the maximum turning angles.



(a) Eight control points to create a cubic Bézier curve

(b) Illustration of points within maximum turning angles

Fig. 1

The two key variables are angle β and length d_0:

$$\beta = \frac{\gamma}{2}, \quad d_0 = \frac{c_4 \cdot \sin \beta}{\kappa_{\max} \cdot \cos^2 \beta} \tag{1}$$

where κ_{max} is the maximum curvature, γ is the angle between the vector $\overrightarrow{P_1P_2}$ and $\overrightarrow{P_2P_3}$, the calculation of c_4 is shown in [18]. Except the initial and goal point, all the extended edges d in our algorithm will be at least two times longer than d_0 to satisfied the robot's allowable maximum angle β and upper bound curvature constraints κ_{max}.

A ball is employed to incorporate neighboring nodes and establish potential new connections to reduce costs in RRT*. For 2D cases, a radius slightly longer than the minimum edge length of the tree (L_{min} or $2d_0$) can be used to define a circle that encompasses potential neighbors x_{nbr}, excluding the current parent node x_{new_p}. Subsequently, RRT* employs two optimization procedures. The first one determines whether to re-select the best candidate tree node within the circle as the parent node for the new node, while the second one determines whether to re-select the new node as the parent node for some or all of its neighbor nodes.

The proposed path planning algorithm aims to find a shorter path $\mathcal{X}_{sol} \subset \mathcal{X}_{free}$ to reach a goal region $\mathbf{X}_{goal}$ within a reasonable time frame while maintaining the feasibility of differential constraints (maximum angles α, curvatures κ, and their continuity) for all path segments. All sets of this path must be continuously connected, bounded, and belong exclusively to the $\mathcal{X}_{free}$ region.

III. PLANNING ALGORITHM

Our proposed algorithm first requires a feasibility check of for a new node or a rewired neighbor node:

- G^2 continuous curvature $\Leftrightarrow B_0X_2 = E_0X_2$
- $\beta(x_1, x_2, x_3) \leq \beta$
- $\kappa_{dir} \leq \kappa_{max} \Leftrightarrow L_{min}$ or $Dist(x_1, x_2) \geq 2d_0$
- graph $\mathcal{T} \subset \mathcal{X}_{free} \Leftrightarrow NoCollisionBézier(x_1, x_2, x_3)$

To satisfy all four constraints, it is essential to guarantee the first one and use function $FeasibleBézier(x_1, x_2, x_3)$ to check the other three. The function $Dist$ calculates the Euclidean distance between two states.

Our algorithm has several key features that improve its effectiveness. First, we add extra kinematic constraints to the two optimization functions from RRT* and ensure proper connection of the points with Bézier curves. Additionally, we incorporate the sampling method from Informed RRT* that directly samples nodes in a prolate hyperspheroid. During the rewiring process of our algorithm, some sub-trees may be discarded or reconnect to other branches when their nodes fail the feasibility check of the kinematic constraints. Although this may affect the previous path that connected to the goal, we still record and utilize the previous path with the current smallest cost towards the goal as the semi-major axis. We use this axis to create an ellipse region that allows for the direct sampling of new potential promising points within it. The recorded smallest cost is crucial because it continues to accelerate the search speed and enables us to find a shorter path. Lastly, we adopted the idea from RRT# to apply heuristic constraints, rewiring only promising nodes. The function $LowerCost(x_1, x_2)$ includes two conditions:

- $c_2 + Dist(x_1, x_2) + Est(x_1, x_{goal}) < c_{best}$
- $c_{new} + Dist(x_1, x_2) \leq c_{nbr}$

To create a piece-wise Bézier curve that satisfies G^2 continuity using Yang's method [18], we split the connections among nodes X_1, X_2, and X_3 into three parts, as shown in Fig 2. Our local spline planner differs slightly from the one described in [17] to solve the two-point boundary problem.

The first and third parts of the curve, (M_2B_0) and (M_1E_0), respectively, are either straight lines or nonexistent. The second part, (B_0E_0), must satisfy the condition that the lengths of B_0X_2 and E_0X_2, which determine 8 control points, are always equal. This condition is crucial because it ensures G^2 continuity. This constraint must be satisfied regardless of whether the curve is created using the steering function or modified by rewire functions of RRT*.

The lengths of B_0X_2 and E_0X_2 can be equal to or larger than the minimum length d_0. If they are equal, the curve may have a larger curvature, as shown in Case 1 in Fig 2. Note that both B_0M_2 and E_0M_1 can be zero.

Alternatively, if the extended edges are slightly larger than d_0, we can utilize the rest of the straight line parts, either M_2B_0 or M_1E_0, to create a Bézier curve with a smaller curvature, as shown in Case 2 in Fig 2. In this case, only one of B_0M_2 or E_0M_1 is set to zero.

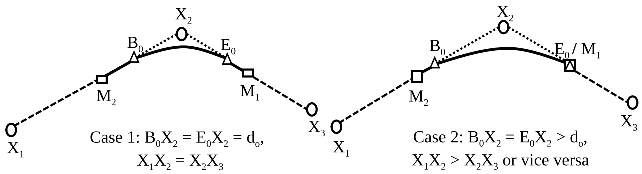


Fig. 2: Methods to connect tree nodes with the new node

Our algorithm builds on the original Informed-RRT* and includes three core functions, as outlined in Algorithm 1. The first function, Algorithm 2 ($Extend$), attaches a new edge from the nearest node, $x_{nearest}$, towards the new point, x_{new}. The second function, Algorithm 3 ($SelectforNew$), re-selects a new parent node from the tree for the new node. Finally, the third function ($SelectforNeighbours$) checks whether a new node can serve as the parent node for its neighbor nodes. We propose three different methods for this function, the last of which is inspired by [16]:

- Trim leaf nodes only
- Partially cut tree branches
- Deleted nodes reuse as "new" sampled nodes

The $SelectForNeighbours$ function is shown in Algorithm 4.1 (Method 1) and Algorithm 4.2 (Method 2 and 3).

Algorithm 1 Cubic Bézier spline Informed RRT*

Input: $V \leftarrow \{x_{start}\}$, $E \leftarrow \emptyset$, $X_{sol} \leftarrow \emptyset, q \leftarrow \emptyset$
Output: $\mathcal{T} = (V, E)$, X_{sol}
  for Iteration$= 1, 2, \ldots, N$ **do**
    $c_{best} \leftarrow min_{x_{sol} \in X_{sol}}\{c_{sol}\}$
    if q is not empty **then**
      $x_{rand} \leftarrow q.pop()$
    else
      $x_{rand} \leftarrow Informed\_Sample(x_{start}, x_{goal}, c_{best})$
    $x_{nearest} \leftarrow (\mathcal{T}, x_{rand})$;
    $x_{new} \leftarrow Extend(\mathcal{T}, x_{nearest}, x_{rand})$
    $X_{nbr} \leftarrow Neighbours(\mathcal{T}, x_{new}, r)$
    $SelectforNew(\mathcal{T}, X_{nbr}, x_{new})$
    if $x_{new} \subsetneq V$ **then**
      $SelectforNeighbours(\mathcal{T}, X_{nbr}, x_{new}, q)$
      if $x_{new} \in X_{goal}$ **then**
        $X_{Sol} \leftarrow X_{Sol} \cup x_{new}$

The first function, $Extend$, is designed to extend the edges of the tree more quickly at the early stage. We treat nodes extended directly from the starting node x_{new} as a unique case. This function first extends the edge from the original point x_{start} to the middle point M_0. This point is located between the origin x_{start} and the new point x_{new}, and the edge is a straight line. Otherwise, we extend the new node using a Bézier curve that connects the parent node of the nearest neighbor x_{nbr_p}, the nearest neighbor node x_{nbr}, and the randomly sampled point x_{rand}. To ensure that the turning angle does not exceed the maximum allowable value β, we calculate the direction from x_{nbr} to x_{rand} using the function $Dir(x_1, x_2)$. If the resulting Bézier curve does not intersect with obstacles in the environment, it is added to the tree. Algorithm 2 shows the details of the $Extend$ function.

Algorithm 2 $Extend(\mathcal{T}, x_{nbr}, x_{rand})$

  $x_{new} \leftarrow x_{nbr} + Dir(x_{nbr}, x_{rand}) * L_{min}$
  if $x_{new\_p} \leftarrow x_{start}$ **then**
    if $CollisionFreeLine(x_{start}, x_{new})$ **then**
      $E \leftarrow E \cup \{x_{start}, M_0\}$ in line; $V \leftarrow V \cup \{x_{new}\}$
  else if $\beta(x_{new}, x_{nbr}, x_{nbr\_p}) < \beta$ and
      $NoCollisionBézier(x_{nbr\_p}, x_{nbr}, x_{new})$ **then**
    $E \leftarrow E \cup \{M_1M_2\}$ in Bézier curve; $V \leftarrow V \cup \{x_{new}\}$

The first part of the rewire process, the function $SelectforNew$ is shown in Algorithm 3, involves several constraints that need to be satisfied before a new node can be added to the tree. Firstly, the new node, the neighbour node, and its parent node must satisfy the maximum turning angle constraint $\beta(x_{new}, x_{nbr}, x_{nbr\_p})$. Secondly, the new edge must be longer than the minimum required length $2d_0$ to avoid exceeding the maximum curvature constraints κ_{max}. Once these constraints are met, the total cost of reaching the new node c_{new} plus the cost of the edge $Dist(x_{new}, x_{nbr})$ must be smaller than the cost-to-come of the neighbour node c_{nbr}. Additionally, the total cost-to-come and estimate-to-go $Est(x_{new}, x_{goal})$ must be smaller than the current smallest cost c_{best}. If all these conditions are met, the new node x_{new} can be added to the tree and connected with a Bézier curve if it is collision-free. As such, this function follows the similar optimization procedure as RRT* but with an additional feasibility check.

Algorithm 3 $SelectForNew(\mathcal{T}, X_{nbr}, x_{new})$

  if x_{new_p} is x_{start} **then** return NULL
  for $\forall x_{nbr} \in X_{nbr}$ **do**
    if x_{nbr} is x_{start} **then**
      if $CollisionFreeLine(x_{start}, x_{new})$ **then**
        $E \leftarrow E \cup \{x_{start}, M_0\}$ with line
        $V \leftarrow V \cup \{x_{new}\}$
        Update c_{new} and c_{best}, break the loop
    else if $LowerCost(x_{new}, x_{nbr})$ and
        $FeasibleBézier(x_{new}, x_{nbr}, x_{nbr\_p})$ **then**
      $E \leftarrow E \cup \{M_1M_2\}$ with Bézier curve
      $V \leftarrow V \cup \{x_{new}\}$
    Update c_{new} and c_{best}

The collision checking strategy needs to consider the piece-wise segments of the Bézier curve, as illustrated in Fig 3. When the new node x_{new} is collision-free, the algorithm attempts to connect it to the nearest node. If the first connection attempt fails, the algorithm will try to connect x_{new} with the other neighbor nodes. However, if both the $Extend$ and $SelectForNew$ functions fail to establish a connection for x_{new}, the sampled node x_{rand} is rejected, and the algorithm starts a new iteration.

The third function, $SelectForNeighbours$, involves three different methods to optimize the RRT tree. The first

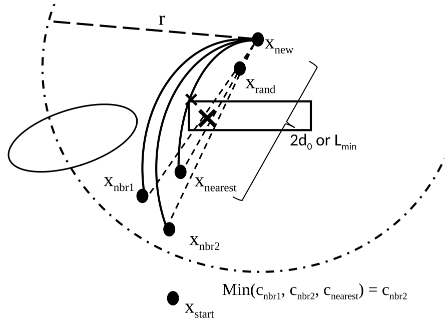


Fig. 3: Collision checking for the initial connection of x_{new}

method simply rejects a neighbour node x_{nbr}, if it has one or more child nodes x_{nbr_cld} and all its x_{succ}). This method provides a quick way to guide the RRT tree to achieve a sub-optimal path towards the final goal region, as it only partially optimizes the leaf nodes. The second method partially prunes branches that are not feasible with the new constraints, which can be effective in reducing the path length. The third method treats these infeasible nodes as "new" sampled nodes, adding them to a queue (q) for the next iteration. If the node is not rejected, the function proceeds to rewire two pairs of edges between three states. The first edge is between the middle points, M_1 and M_2, of neighbour node x_{nbr}, new node x_{new}, and its parent node x_{new_p}. The second edge is between the middle points, M_1 and M_2, of the new node x_{new}, neighbour node x_{nbr}, and all of its child nodes x_{nbr_cld}. The illustration is shown in Fig 4. The $SelectForNeighbours$ function is shown in Algorithm 4.1 (Method 1) and Algorithm 4.2 (Method 2 and 3).

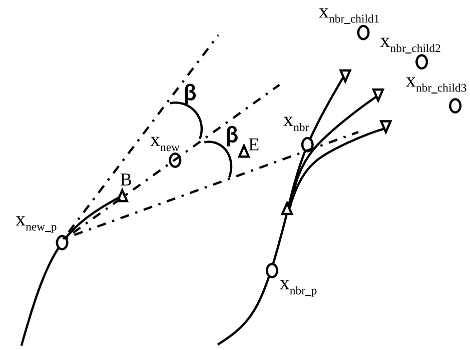
Algorithm 4.1 $SelectForNeighbours(\mathcal{T}, X_{nbr}, x_{new}, q)$

for $\forall x_{nbr} \in X_{nbr}$ **do**
  if x_{nbr_cld} exists **then**
    Skip this iteration
  if $LowerCost(x_{nbr}, x_{new})$ and $FeasibleBézier(x_{nbr}, x_{new}, x_{new\_p})$ **then**
      $E \leftarrow E \cup \{M_1 M_2\}$ with Bézier curve
    Update c_{nbr} and c_{best}

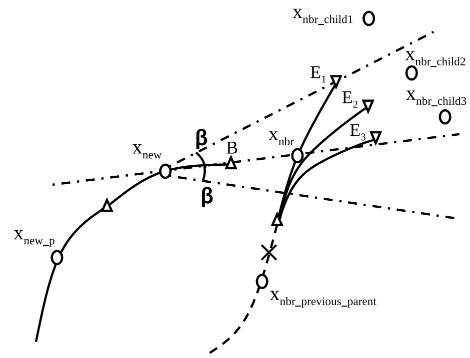
Algorithm 4.2 $SelectForNeighbours(\mathcal{T}, X_{nbr}, x_{new}, q)$

for $\forall x_{nbr} \in X_{nbr}$ **do**
  if $LowerCost(x_{nbr}, x_{new})$ and $FeasibleBézier(x_{nbr}, x_{new}, x_{new\_p})$ **then**
    $E \leftarrow E \cup \{M_1 M_2\}$ with Bézier curve
    if x_{nbr_cld} exists **then**
      if $FeasibleBézier(x_{new}, x_{nbr}, x_{nbr\_cld})$ **then**
        $E \leftarrow E \cup \{M_1 M_2\}$ with Bézier curve
      else
        Remove x_{nbr_cld} and all x_{succ} **(Method 2)**
        Put x_{nbr_cld} and all x_{succ} into q **(Method 3)**
    Update c_{nbr}, all its c_{nbr_cld} and c_{best}.

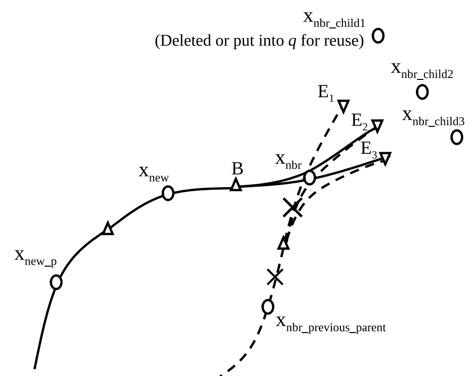
One important aspect in the algorithm is the calculation of the edge length when a new node is added to the tree. One simple solution is to use straight-lines connected among all sampled points that create a Bézier curve as the length for the cost. However, to ensure that the ranges of the ellipse cover enough sampling space, this method requires additional cost. To obtain a more accurate calculation of the length of each Bézier curve segment, advanced numerical methods can be used, as there is no closed-form solution for the length of cubic or higher-order Bézier curves.



(a) Feasibility check among $x_{nbr}, x_{new}, x_{new\_p}$



(b) Feasibility check among $x_{nbr}, x_{new}, x_{nbr\_cld}$



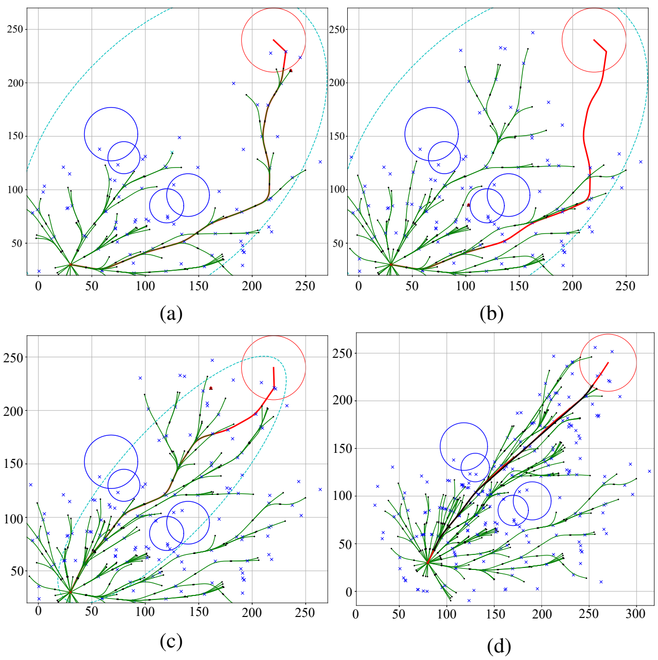
(c) Some x_{nbr_cld} retained, some x_{nbr_cld} with its x_{succ} deleted

Fig. 4: The second rewire process

IV. Simulation Results

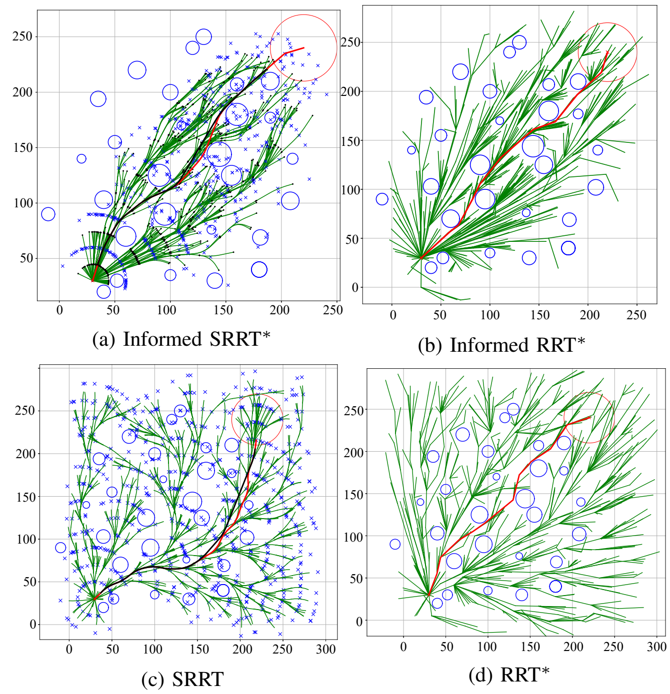
We have developed an optimal cubic Bézier spline RRT algorithm that incorporates kinematic constraints in Python 3. Simulation 1, depicted in Fig 5, is primarily intended for illustrative purposes. In Simulation 2, we conducted an extensive run of 1000 iterations and compared the results to SRRT, RRT*, and informed RRT*.

The map measures $280m$ by $280m$ and is limited to the



(a) (b) (c) (d)

Fig. 5: The simulation of Informed SRRT* with a narrow passage



(a) Informed SRRT* (b) Informed RRT* (c) SRRT (d) RRT*

Fig. 6: Comparison among Informed SRRT*, Informed RRT*, SRRT and RRT* with massive clustered obstacles

first quadrant $(x, y > 0)$. The goal region in simulations is a circle with a radius of $30m$ centered at $(220m, 240m)$. The starting point is set at $(30m, 30m)$. The maximum curvature κ_{max}, angle 2β and the minimum extended length $2d_0$ is set as 0.1, 0.4π and $20.18m$, respectively, as suggested in [17]. For simplicity, our extended length L_{min} is fixed as $30m$ and our radius r in 2D environment is fixed as $45m$. In the figures, the blue cross sign denotes nodes that have successfully connected to the tree as long as the edges have no collision with obstacles, while blue circles represent obstacles. The green curves represent the edges of the tree, and black dots indicate the connection points among edges. Due to the map's resolution of 0.5, some tiny gaps are acceptable between two black dots. The red path represents the final path, with a cost close to the optimal region $[Est(x_{start}, x_{goal}) \pm D_{goal}]$. We also applied path smoothing techniques to generate the black path.

Figure 6 illustrates the comparison among Informed SRRT* and three other conventional algorithms. We conducted 20 repeated experiments to compare these four algorithms, and the average final path lengths, range, and program running times are presented in Table I. Comparative results can be found in Figure 7.

	a	b	c	d
Range(s)	[737.49, 1001.06]	[162.37, 184]	[267.19, 327.16]	[184.38, 205.67]
T_{avg}(s)	892.29	174.01	300.58	190.95
L_{avg}(m)	294.3	291.03	322.96	297.11

TABLE I: a: Informed SRRT*, b: Informed RRT*, c: SRRT, d: RRT*

V. Conclusions and Future Work

In this paper, we have introduced a novel sampling-based algorithm that optimizes the length of the final path towards the goal while considering kinematic constraints. These constraints encompass acceptable turning angles and G^2 continuity of the curvature. Our algorithm can effectively avoid obstacles and extend as piece-wise segments.

During the initial steering process, we accept a newly sampled point even if it cannot immediately connect to the nearest point due to path collision. Our optimization function consists of two main parts. In the first part, we aim to connect the new sampled node to the best neighbor node that satisfies kinematic constraints, ensuring collision-free paths. The selected neighbor is determined based on the shortest estimated path length to the goal region, along with the cost from the starting point. In the second part, we propose three different methods. The first method is effective in finding an initial feasible path towards the goal. The second and third methods have the ability to rewire tree nodes when at least three consecutive nodes satisfy kinematic constraints. Through simulation results, we have demonstrated the correctness and effectiveness of the modified optimization processes from RRT* and achieved similar smoothness like SRRT. Lastly, the path smoothing techniques further improved the final path.

In addition to the aforementioned contributions, this method can be extended to informed SRRT#. Unlike RRT* using B-splines, our method can pass through the sampled local control nodes. Going forward, we are interested in exploring other spline methods and extending this approach to non-Euclidean state spaces.

Acknowledgement

This work is funded by the Mitacs Accelerate program of the project named "Intelligent Unmanned Surface Vehicles with Safe Navigation and Docking in Harsh Marine Environ-

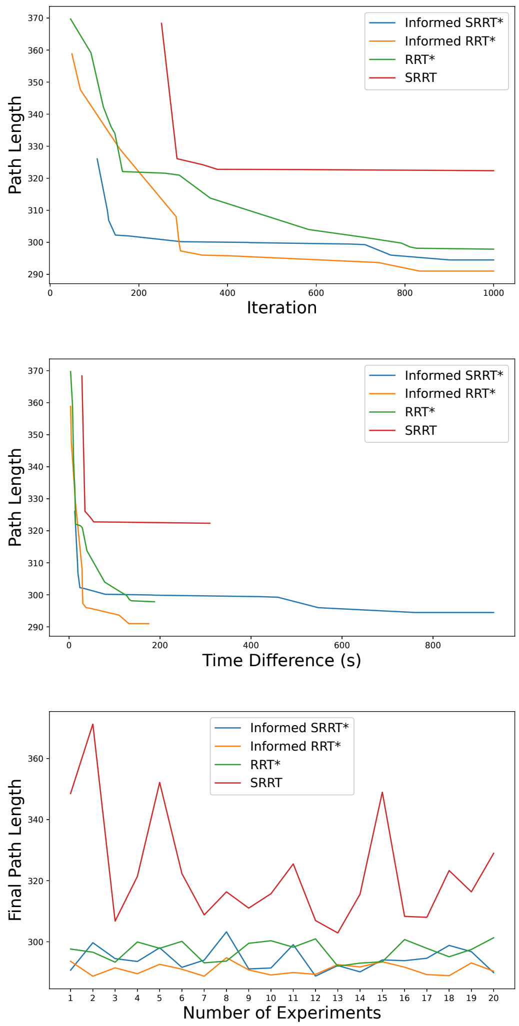


Fig. 7: Analytical comparison of four RRT algorithms

ment” in collaboration with Marine Thinking Inc, Canada. Special thanks to all previous and current members in USV team of Marine Thinking Inc and all reviewers’ comments.

References

[1] D. Dolgov and S. Thrun, “Autonomous driving in semi-structured environments: Mapping and planning,” in *2009 IEEE international conference on robotics and automation*, IEEE, 2009, pp. 3407–3414.

[2] A. Vagale, R. Oucheikh, R. T. Bye, O. L. Osen, and T. I. Fossen, “Path planning and collision avoidance for autonomous surface vehicles i: A review,” *Journal of Marine Science and Technology*, pp. 1–15, 2021.

[3] X. Zhou, Z. Wang, H. Ye, C. Xu, and F. Gao, “EGO-Planner: An ESDF-Free gradient-based local planner for quadrotors,” *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 478–485, 2021.

[4] J.-C. Latombe, *Robot motion planning*. Springer Science & Business Media, 2012, vol. 124.

[5] M. Shomin and R. Hollis, “Fast, dynamic trajectory planning for a dynamically stable mobile robot,” in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2014, pp. 3636–3641.

[6] C. Richter, A. Bry, and N. Roy, “Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments,” in *Robotics research*, Springer, 2016, pp. 649–666.

[7] O. Arslan and P. Tsiotras, “The role of vertex consistency in sampling-based algorithms for optimal motion planning,” *arXiv:1204.6453*, 2012.

[8] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, “Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic,” in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2014, pp. 2997–3004.

[9] M. Lichocki and L. Rodrigues, “A gaussian-biased heuristic for stochastic sampling-based 2d trajectory planning algorithms,” in *2020 European Control Conference (ECC)*, IEEE, 2020, pp. 1949–1954.

[10] M. W. Otte and E. Frazzoli, “RRT^X: Real-time motion planning/replanning for environments with unpredictable obstacles,” in *WAFR*, 2014.

[11] D. Connell and H. M. La, “Dynamic path planning and replanning for mobile robots using RRT*,” in *2017 IEEE International Conference on Systems, Man, and Cybernetics*, IEEE, 2017, pp. 1429–1434.

[12] S. Koenig and M. Likhachev, “Fast replanning for navigation in unknown terrain,” *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 354–363, 2005.

[13] S. M. LaValle *et al.*, “Rapidly-exploring random trees: A new tool for path planning,” 1998.

[14] S. Karaman and E. Frazzoli, “Sampling-based algorithms for optimal motion planning,” *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[15] F. Islam, J. Nasir, U. Malik, Y. Ayaz, and O. Hasan, “RRT*-Smart: Rapid convergence implementation of RRT* towards optimal solution,” in *2012 IEEE international conference on mechatronics and automation*, IEEE, 2012, pp. 1651–1656.

[16] G. Vailland, V. Gouranton, and M. Babel, “Cubic bézier local path planner for non-holonomic feasible and comfortable path generation,” in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 7894–7900.

[17] K. Yang *et al.*, “Spline-based RRT path planner for non-holonomic robots,” *Journal of Intelligent & Robotic Systems*, vol. 73, no. 1, pp. 763–782, 2014.

[18] K. Yang and S. Sukkarieh, “An analytical continuous-curvature path-smoothing algorithm,” *IEEE Transactions on Robotics*, vol. 26, no. 3, pp. 561–568, 2010.